SANTANDER CUSTOMER TRANSACTION PREDICTION

Venkata Sai Maneesha Gara

23 September 2019

Table of Contents

List of figures

Introduction

1. Problem Statement

The objective of this case is to help people and businesses prosper based on Anonymous Customer Satisfaction data of Santander Bank provided from which we can identify is a customer satisfied? Will a customer buy this product? Can a customer pay this loan?

The details of data attributes in the dataset are as follows –

Variable	description
target	Categorical variable having 0 or 1 which indicates customer transaction not done or done respectively.
ID_code	It's a string variable , which is transaction ID.
Var_0 to Var_199	Different variables of anonymous data which are of numeric data type.

2. Sample Data set

Below is the structure of the data. 200000 obeservation and 202 columns with object and float64 as a datatype.

ID_code	target	var_0	var_1	var_2	var_3	var_4	var_5	var_6	var_7	...	var_190	var_191	var_192	var_193	var_194	var_195	var_196	vai
train_0	0	8.9255	-6.7863	11.9081	5.0930	11.4607	-9.2834	5.1187	18.6266	...	4.4354	3.9642	3.1364	1.6910	18.5227	-2.3978	7.8784	8.
train_1	0	11.5006	-4.1473	13.8588	5.3890	12.3622	7.0433	5.6208	16.5338	...	7.6421	7.7214	2.5837	10.9516	15.4305	2.0339	8.1267	8.
train_2	0	8.6093	-2.7457	12.0805	7.8928	10.5825	-9.0837	6.9427	14.6155	...	2.9057	9.7905	1.6704	1.6858	21.6042	3.1417	-6.5213	8.
train_3	0	11.0604	-2.1518	8.9522	7.1957	12.5846	-1.8361	5.8428	14.9250	...	4.4666	4.7433	0.7178	1.4214	23.0347	-1.2706	-2.9275	10.
train_4	0	9.8369	-1.4834	12.8746	6.6375	12.2772	2.4486	5.9405	19.2514	...	-1.4905	9.5214	-0.1508	9.1942	13.2876	-1.5121	3.9267	9.

Fig1.1. first 5 observations

Data Preprocessing and Exploratory Data Analysis

Any predictive modeling requires that we look at the data before we start modeling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis.

1. TypeConversion

To perform any techniques , since the datatype of all the anonymous data is in float we donot have to convert anything.if the target variable is of categorical data type then that has to be changed to numeric.

2. Missing Value Analysis

To perform a missing value we have different statistical methods like mean , mode , median and advance data mining methods like KNN imputation. Check for the best approach to fillup the missing values. The dataset provided doesnot contain any missing values.

3. Removal of Constants

There is no purpose of handling constants in the dataset as they doesnot have any relationship with the target variable since they are constants.

Remove all the constant values from the training dataset before you proceed to feature engineering

```
There are no missing values present
missing values train count :  0
There are no missing values present
missing values test count :  0
There are no constant present
constant values count :  0
There are no constant present
constant values count :  0
```

4. Check for the balance of data

Based on the training data we can predict tha only 10% of the customers are satisfied out of 200000 transactions.

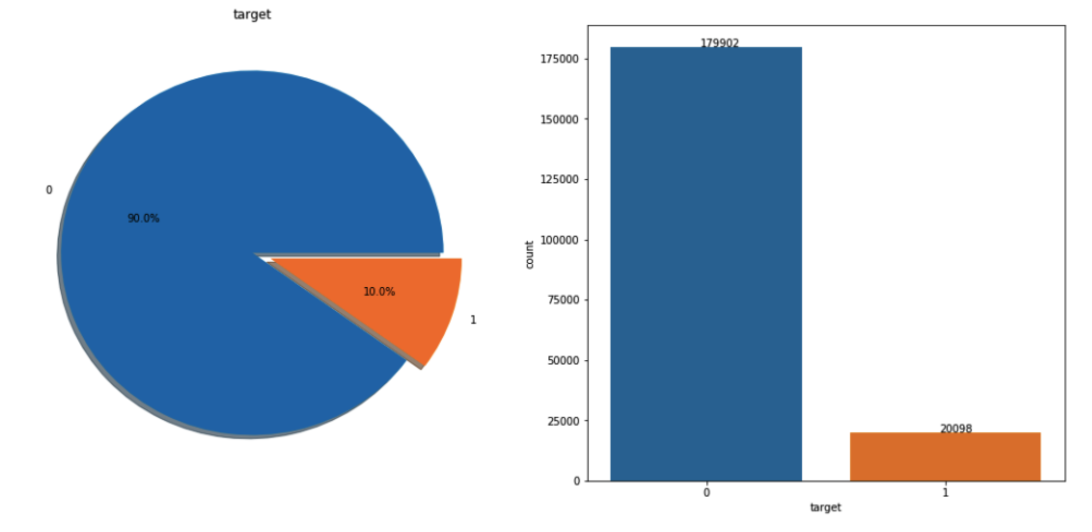

Fig2.1. Summary of target variable balance in the train dataset

5. Feature Importance

Out of 200 features the below are the 20 features which has top most significance on the target variable.

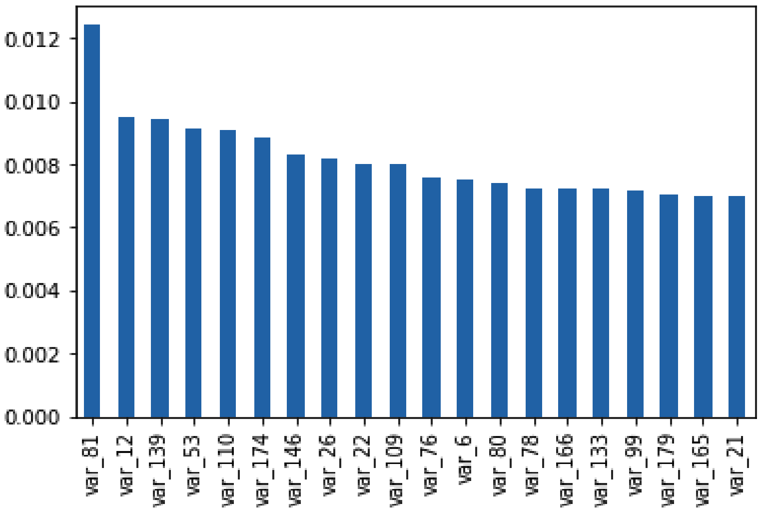

Fig2.2. Feature Importance

6. Visualizations

a. Density of target variables values for the important features

The below density curves indicates that distribution of var values when target value is 1 and target values is 0.

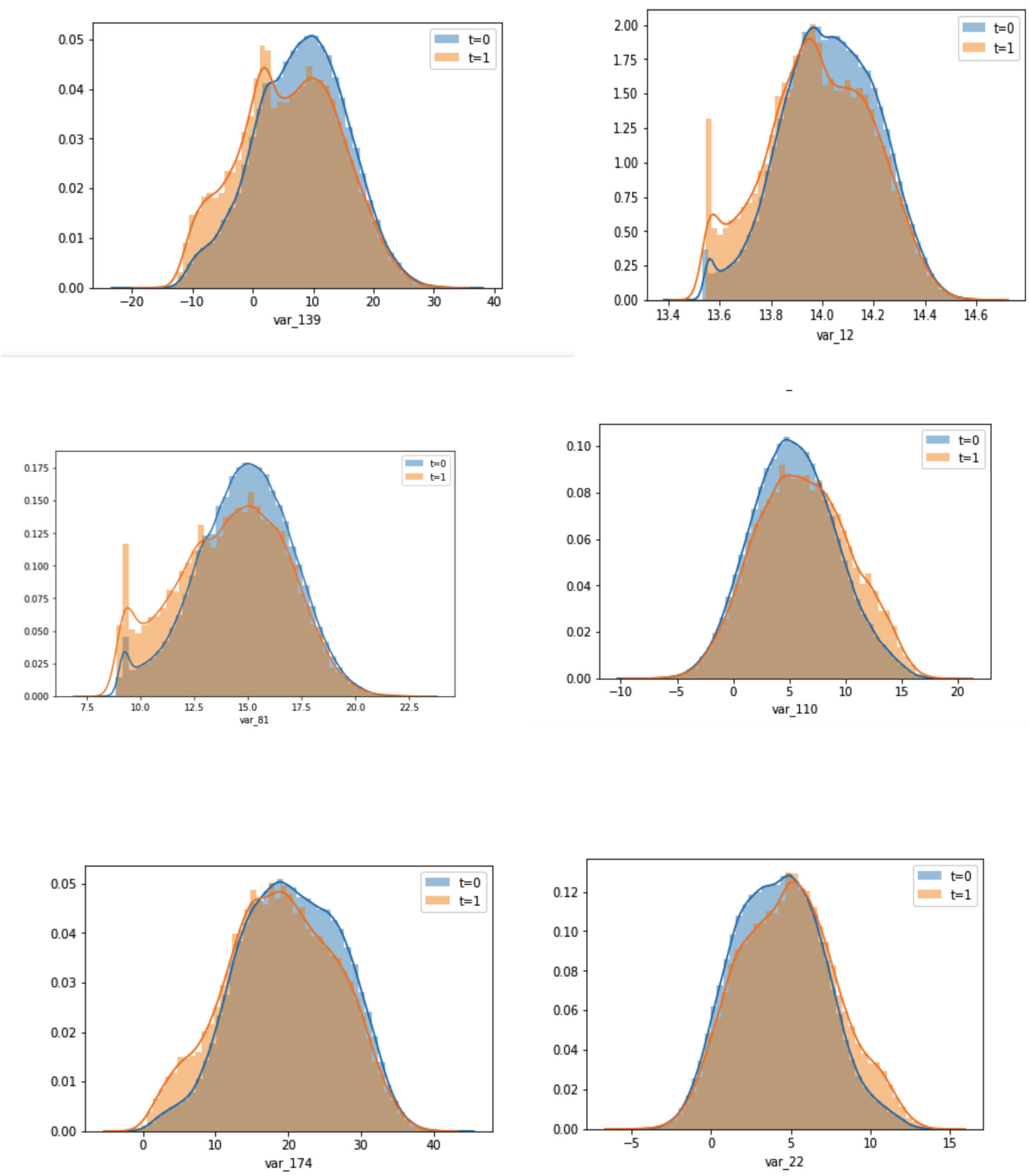

Fig2.3. Density of results by features

b. Correlation Analysis

From below visualization,there is no corelation between the features. Each variable is independent to each other.

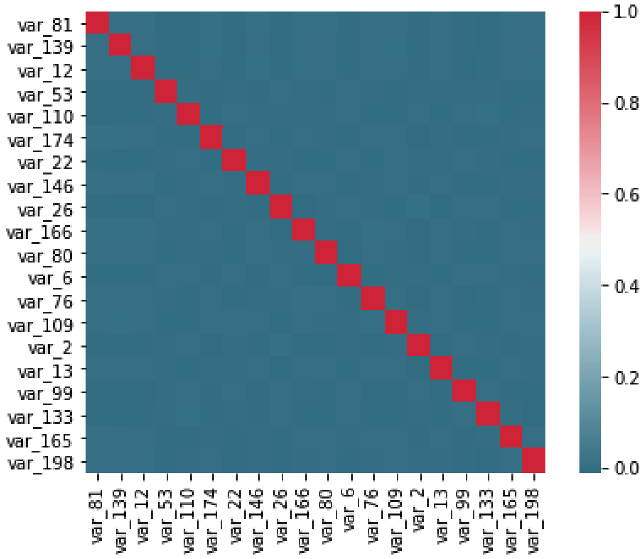

Fig2.4.Correlation Analysis

Model Development

1. Logistic Regression

Logistic regression is the appropriate regression analysis to conduct when the dependent variable is dichotomous (binary). Like all regression analyses, the logistic regression is a predictive analysis. Logistic regression is used to describe data and to explain the relationship between one dependent binary variable and one or more nominal, ordinal, interval or ratio-level independent variables.

2. Decision Tree Classifier

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm. The time complexity of decision trees is a function of the number of records and number of attributes in the given data. The decision tree is a distribution-free or non-parametric method, which does not depend upon probability distribution assumptions. Decision trees can handle high dimensional data with good accuracy.

3. Naive Bayes Classifier

Naive Bayes classifiers are a collection of classification algorithms based on **Bayes' Theorem**. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

4. Random Forest Classifier

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement if bootstrap=True

5. XGBoost Classifier

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. The same code runs on major distributed environment (Hadoop, SGE, MPI)

Model Validation

The model validation can be done by calculating various error metrics. Since it is a Classification problem. The major metrics would be -

- Confusion Matrix (Accuracy, Precision , Sensitivity or Recall , Specificity)
- AUC

i. Confusion Matrix

A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. It allows the visualization of the performance of an algorithm.

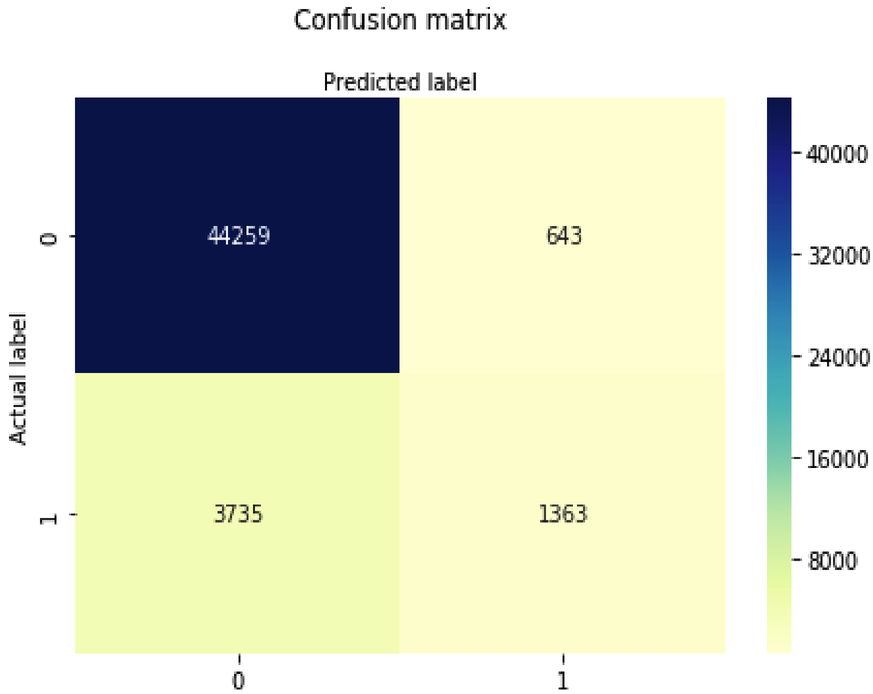

Fig3.1.Confusion Matrix

ii. ROC Curve

Receiver Operating Characteristic curve, or ROC curve, is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The ROC curve is created by plotting the true positive rate against the false positive rate at various threshold settings.

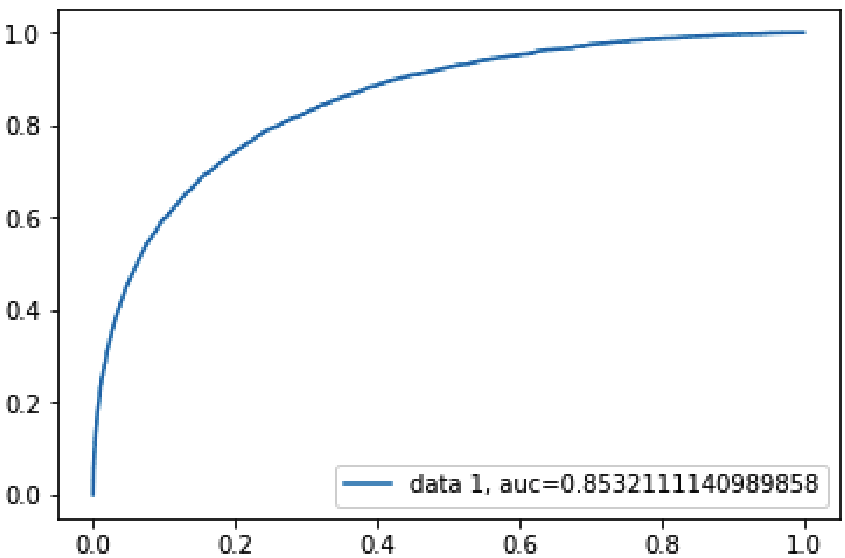

Fig3.2.ROC Curve

	RandomForest	NaviesBayes	DecisionTree	LogisticRegression
Accuracy	0.900900	0.921150	0.836150	0.912440
Precision	0.692308	0.698900	0.189306	0.679462
FalsePositiveRate	0.000222	0.017228	0.093433	0.014320
Specificity	0.999778	0.982772	0.906567	0.985680
Recall or Sensitivity	0.004529	0.362523	0.197786	0.267360
ROC	0.502154	0.672647	0.552177	0.626520

Fig3.3.Error Metrics

Conclusion

- Based on the error analysis ,we conclude that on applying Random Forest and XGBoost give us more accurate result than DecisionTree , LogisticRegression , NaviesBayes
- Once the model is fixed we can now predict for the any test data provided , whether a customer has done the transaction or not. PFB summary of test values predicted.

var_2	var_3	var_4	var_5	var_6	var_7	var_8	var_9	...	var_192	var_193	var_194	var_195	var_196	var_197	var_198	var_199	target	ID
12.9536	9.4292	11.4327	-2.3805	5.8493	18.2675	2.1337	8.8100	...	-1.4300	2.4508	13.7112	2.4669	4.3654	10.7200	15.4722	-8.7197	0	test_0
11.3047	5.1858	9.1974	-4.0117	6.0196	18.6316	-4.4131	5.9739	...	0.9403	10.1282	15.5765	0.4773	-1.4852	9.8714	19.1293	-20.9760	0	test_1
10.1407	7.0479	10.2628	9.8052	4.8950	20.2537	1.5233	8.3442	...	1.9803	2.1800	12.9813	2.1281	-7.1086	7.0618	19.8956	-23.1794	0	test_2
12.0220	6.5749	8.8458	3.1744	4.9397	20.5660	3.3755	7.4578	...	1.6580	3.5813	15.1874	3.1656	3.9567	9.2295	13.0168	-4.2108	0	test_3
14.1295	7.7506	9.1035	-8.5848	6.8595	10.6048	2.9890	7.1437	...	1.2835	3.3778	19.5542	-0.2860	-5.1612	7.2882	13.9260	-9.1846	0	test_4

APPENDIX

Python-Code

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
#Import Libraries for decision tree
from sklearn import tree
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LogisticRegression
import xgboost as xgb


os.getcwd()
os.chdir('/Users/maneeshagvs/Documents/datasets')
train = pd.read_csv('Train_dataset.csv')
test = pd.read_csv('Test_dataset.csv')
test_id = test.ID_code
Y =train['target']
X = train.drop(columns=['target','ID_code'])
test = test.drop(columns=['ID_code'])
error_table = pd.DataFrame()
def missingvalues(df):
   count = 0
   for i in df.columns:
      if(df[i].isnull().sum() != 0):
         print("The missing values are present for",df[i].isnull().sum(),i)
         count+=1
   if(count == 0):
      print("There are no missing values present")
   return count

def constants(df):
```

```
    count = 0
    for i in df.columns:
        if(len(df[i].unique()) == 1):
            print("The value is constant",len(df[i].unique()),i)
            count+=1
    if(count == 0):
        print("There are no constant present")
    return count

print("missing values train count : ",missingvalues(train))
print("missing values test count : ",missingvalues(test))
print("constant values count : ",constants(train))
print("constant values count : ",constants(test))


def Remove_duplicate():
    remove = []
    cols = train.columns
    for i in range(len(cols)-1):
        v = train[cols[i]].values
    for j in range(i+1,len(cols)):
        if np.array_equal(v,train[cols[j]].values):
            remove.append(cols[j])
    train.drop(remove, axis=1, inplace=True)
    test.drop(remove, axis=1, inplace=True)

def accuracy_cal(i,actual,predicted):
    CM = pd.crosstab(actual,predicted)
    #let us save TP, TN, FP, FN
    TN = CM.iloc[0,0]
    FN = CM.iloc[1,0]
    TP = CM.iloc[1,1]
    FP = CM.iloc[0,1]
    error_table.loc['Accuracy',i]=(((TP+TN))/(TP+TN+FP+FN))
    error_table.loc['Precision',i]=((TP)/(TP+FP))
    error_table.loc['FalsePositiveRate',i]=((FP)/(FP+TN))
    error_table.loc['Specificity',i]= 1 - error_table.loc['FalsePositiveRate',i]
    error_table.loc['Recall or Sensitivity',i]=((TP)/(TP+FN))
    print("Accuracy ,
FalsePositive:",error_table.loc['Accuracy',i],error_table.loc['FalsePositiveRate',i])
    auc = metrics.roc_auc_score(actual,predicted)
    error_table.loc['ROC',i]=auc

Remove_duplicate()

train0 = train[ train['target']==0 ].copy()
train1 = train[ train['target']==1 ].copy()
```

In[3]:

```
f,ax=plt.subplots(1,2,figsize=(18,8))
train['target'].value_counts().plot.pie(explode=[0,0.1],autopct='%1.1f%%',ax=ax[0],sh
adow=True)
ax[0].set_title('target')
ax[0].set_ylabel('')
s = sns.countplot(train["target"],
              order = train["target"].value_counts().index)
for p, label in zip(s.patches, train["target"].value_counts()):
    s.annotate(label, (p.get_x()+0.375, p.get_height()+0.15))

plt.show()
```

In[60]:

```
train.head()
```

In[61]:

```
X1_train, X1_test, Y1_train, Y1_test = train_test_split(X, Y, test_size=0.3,
random_state=40)
RF_model = RandomForestClassifier(n_estimators = 20).fit(X1_train, Y1_train)
RF_Predictions = RF_model.predict(X1_test)
accuracy_cal('RandomForest',Y1_test,RF_Predictions)
```

In[62]:

```
train.info()
```

In[63]:

```
(pd.Series(RF_model.feature_importances_,
index=X.columns).nlargest(20).plot(kind='bar'))
```

In[64]:

```
feature_importance = pd.DataFrame(pd.Series(RF_model.feature_importances_,
index=X.columns).nlargest(20))
train_imp = train.loc[:,feature_importance.index]
f, ax = plt.subplots(figsize=(7, 5))

#Generate correlation matrix
corr = train_imp.corr()

#Plot using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),
cmap=sns.diverging_palette(220, 10, as_cmap=True),
        square=True, ax=ax)


# In[65]:


plt.figure(figsize=(8,5))
for i in train_imp.columns:
   sns.distplot(train0[i], label = 't=0')
   sns.distplot(train1[i], label = 't=1')
   plt.legend()
   plt.xlabel(i)
   plt.show()


# In[66]:


train[train.columns[2:]].mean().plot('hist');plt.title('Mean Frequency');


# In[67]:


######################################## NAIVE_BAYES
##############################################################
#######
#Naive Bayes
from sklearn.naive_bayes import GaussianNB
#Naive Bayes implementation
X2_train, X2_test, Y2_train, Y2_test = train_test_split(X, Y, test_size=0.3,
random_state=40)

NB_model = GaussianNB().fit(X2_train, Y2_train)

#predict test cases
NB_Predictions = NB_model.predict(X2_test)
```

```
accuracy_cal('NaviesBayes',Y2_test,NB_Predictions)


# In[68]:


#################################### DECISION TREE
#############################################################
######
#Decision Tree
X4_train, X4_test, Y4_train, Y4_test = train_test_split(X, Y,
test_size=0.3,random_state=40)

C50_model = tree.DecisionTreeClassifier(criterion='entropy').fit(X4_train, Y4_train)

#predict new test cases
C50_Predictions = C50_model.predict(X4_test)
accuracy_cal('DecisionTree',Y4_test,C50_Predictions)


# In[69]:


# split X and y into training and testing sets

X5_train,X5_test,Y5_train,Y5_test=train_test_split(X,Y,test_size=0.25,random_state=0)


# In[70]:


################################## LOGISTIC REGRESSION
#############################################################
##

# import the class
from sklearn.linear_model import LogisticRegression

# instantiate the model (using the default parameters)
logreg = LogisticRegression()

# fit the model with data
logreg.fit(X5_train,Y5_train)

#
Y5_pred=logreg.predict(X5_test)

accuracy_cal('LogisticRegression',Y5_test,Y5_pred)
```

```
# In[71]:


cnf_matrix = confusion_matrix(Y5_test,Y5_pred)


# In[72]:


class_names=[0,1] # name  of classes
fig, ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
# create heatmap
sns.heatmap(pd.DataFrame(cnf_matrix), annot=True, cmap="YlGnBu" ,fmt='g')
ax.xaxis.set_label_position("top")
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')


# In[73]:


y_pred_proba = logreg.predict_proba(X5_test)[::,1]
fpr, tpr, _ = metrics.roc_curve(Y5_test,  y_pred_proba)
auc = metrics.roc_auc_score(Y5_test, y_pred_proba)
plt.plot(fpr,tpr,label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()


# In[74]:


X1_train, X1_test, Y1_train, Y1_test = train_test_split(X, Y, test_size=0.3,
random_state=40)
## # Feature selection
clf = ExtraTreesClassifier(random_state=1729)
selector = clf.fit(X1_train, Y1_train)


# In[75]:
```

```
#####################################XGBOOST
##############################################################
################

fs = SelectFromModel(selector, prefit=True)
X1_train = fs.transform(X1_train)
X1_test = fs.transform(X1_test)
test = fs.transform(test)

print(X1_train.shape, X1_test.shape, test.shape)

## # Train Model
# classifier from xgboost
m2_xgb = xgb.XGBClassifier(n_estimators=110, nthread=-1, max_depth =
4,seed=1729)
m2_xgb.fit(X1_train,Y1_train, eval_metric="auc", verbose = False,eval_set=[(X1_test,
Y1_test)])

# calculate the auc score
print("Roc AUC:", metrics.roc_auc_score(Y1_test, m2_xgb.predict_proba(X1_test)[:,1],
          average='macro'))

auc = metrics.roc_auc_score(Y1_test, m2_xgb.predict_proba(X1_test)[:,1])
plt.plot(fpr,tpr,label="data 1,auc="+str(auc))
plt.legend(loc=4)
plt.show()


# In[80]:


accuracy_cal('XGBoost',Y1_test,m2_xgb.predict_proba(X1_test)[:,1])


# In[76]:


## final Submission
probs = m2_xgb.predict_proba(test)
submission = pd.DataFrame({"ID":test_id, "TARGET": probs[:,1]})
submission.to_csv("submission.csv", index=False)


# In[85]:


###over error metrics
print(error_table)
```

R-Code

```
rm(list = ls())
getwd()
setwd("~/Documents/datasets")
#################################### load library files
 ##################################################################
x =
 c("geosphere","stringr","DMwR","caret","rpart","MASS","usdm",'randomForest','sqldf','ggplot2'
 ,"xgboost",'Matrix','pie3D',"C50", "dummies", "e1071", "Information",
    "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees')

lapply(x, require, character.only = TRUE)
rm(x)
####################################load library files
 ##################################################################
train1 = read.csv("Train_dataset.csv", header = TRUE)
test1 = read.csv("Test_dataset.csv", header = TRUE)
train = train1[,-c(1)]
test = test1[,-c(1)]
target = train$target
test_ID = test$ID_code

(t <- table(train$target) / nrow(train))
require(plotrix)
l <- paste(c('Happy customers\n','Unhappy customers\n'), paste(round(t*100,2), '%', sep=''))
pie3D(t, labels=l, col=c('green','red'), main='Santander customer satisfaction dataset',
 theta=1, labelcex=0.8)


############################### DATA PREPROCESSING
 ###################################################################
 ###############
# remove constant features
for (i in names(train)) {
 if (length(unique(train[[i]])) == 1) {
   cat(i, "is constant in train (", unique(train[[i]]),"). We delete it.\n")
   train[[i]] <- NULL
   test[[i]] <- NULL
 }
}

# remove identical features
features_pair <- combn(names(train), 2, simplify = F)
toRemove <- c()
for(pair in features_pair) {
 f1 <- pair[1]
 f2 <- pair[2]

 if (!(f1 %in% toRemove) & !(f2 %in% toRemove)) {
   if (all(train[[f1]] == train[[f2]])) {
```

```
      cat(f1, "and", f2, "are equal.\n")
      toRemove <- c(toRemove, f2)
    }
  }
}

feature.names <- setdiff(names(train[,-c(1)]), toRemove)

train <- train[,feature.names]
test <- test[,feature.names]

# Removing highly correlated variables
cor_v <- abs(cor(train))
diag(cor_v) <- 0
cor_v[upper.tri(cor_v)] <- 0
cor_f <- as.data.frame(which(cor_v > 0.85, arr.ind = TRUE))
#train <- train[,-unique(cor_f$row)]
#test <- test[,-unique(cor_f$row)]

image(cor_v)

anyNA(train)
anyNA(test)
summary(train)
##############################  MODEL DEVELOPMENT
 ######################################################################
 ###########
train$target = target
train$target = as.factor(train$target)
#Divide data into train and test using stratified sampling method
set.seed(1234)
train.index = createDataPartition(train$target, p = .80, list = FALSE)
Train2 = train[ train.index,]
Test2 = train[-train.index,]

##Decision tree for classification
#Develop Model on training data
#C50_model = C5.0(target ~., Train2, trials = 10, rules = TRUE)

#Summary of DT model
summary(C50_model)

#write rules into disk
#write(capture.output(summary(C50_model)), "c50Rules.txt")

#Lets predict for test cases
#C50_Predictions = predict(C50_model,subset(mydata, select = -c(target)), type = "class")

##Evaluate the performance of classification model
#ConfMatrix_C50 = table(test$target, C50_Predictions)
#confusionMatrix(ConfMatrix_C50)

#Accuracy: 90.89%
#FNR: 63.09%
```

```
########################################################## END OF DECISION
 TREES #######################################################

###Random Forest
#RF_model = randomForest(target ~ ., Train2, importance = TRUE, ntree = 2)

#Presdict test data using random forest model
#RF_Predictions = predict(RF_model,subset(Train2, select = -c(target)))

##Evaluate the performance of classification model
#ConfMatrix_RF = table(Test2$responded, RF_Predictions)

#print(confusionMatrix(ConfMatrix_RF))

#Accuracy: 93.89%
#FNR: 58.09%

################################################# END OF RANDOM FOREST
 ###############################################################
library('usdm')

#Logistic Regression
logit_model = glm(target ~ ., data = Train2, family = "binomial")

#summary of the model
summary(logit_model)

#predict using logistic regression
logit_Predictions = predict(logit_model, newdata = Test2, type = "response")

#convert prob
logit_Predictions = ifelse(logit_Predictions > 0.5, 1, 0)


##Evaluate the performance of classification model
ConfMatrix_log = table(Test2$target, logit_Predictions)
print(confusionMatrix(ConfMatrix_log))

#accuracy - 91.4
#FPR - 32.2

######################################## END OF LOGISTIC REGRESSION
 ###################################################################
#naive Bayes
library(e1071)

#Develop model
NB_model = naiveBayes(target ~ ., data = Train2)

#predict on test cases #raw
NB_Predictions = predict(NB_model,subset(Test2, select = -c(target)), type = 'class')
```

```
#Look at confusion matrix
Conf_matrix = table(Test2$target,predicted = NB_Predictions)
print(confusionMatrix(Conf_matrix))

#Accuracy: 92.16
#FPR: 29.57
##################################################### END OF NAIVE
 BAYES ###########################################################

logit_Predictions = predict(logit_model, newdata = test, type = "response")

#convert prob
test$target = ifelse(logit_Predictions > 0.5, 1, 0)

write.csv(test,"submission_R.csv")
```

End of the R Code